Mini Projet ML

Omar Killbi

MPDS1

Partie 1 :

Question 1 : préparation du fichier Arrf

Un fichier *arff* comprend toujours trois types d'informations : un nom pour la base de données, des attributs et des données. La chaîne de caractères `@RELATION` permet de donner un nom à la base de données. Par exemple, dans le cas du fichier *iris.arff*, le nom donné est *iris*.

```
@RELATION iris
```

Dans notre cas et selon le fichier « caracteristiquevehicules.csv » on a 8 attributs :

- Identifiant
- Carrosserie
- Empattement
- Longueur
- Nombre-cylindres
- Puissance
- KilometrageMoyen
- Prix

Figure 1: fichier arff

Après la préparation, on va importer le fichier .arff sur le weka.

Question 2 :

Dans cette partie on va planifier le prétraitement nécessaire sur les données avec la méthode de filtre non supervisé pour l'attribut prix comme montre la figure 2.

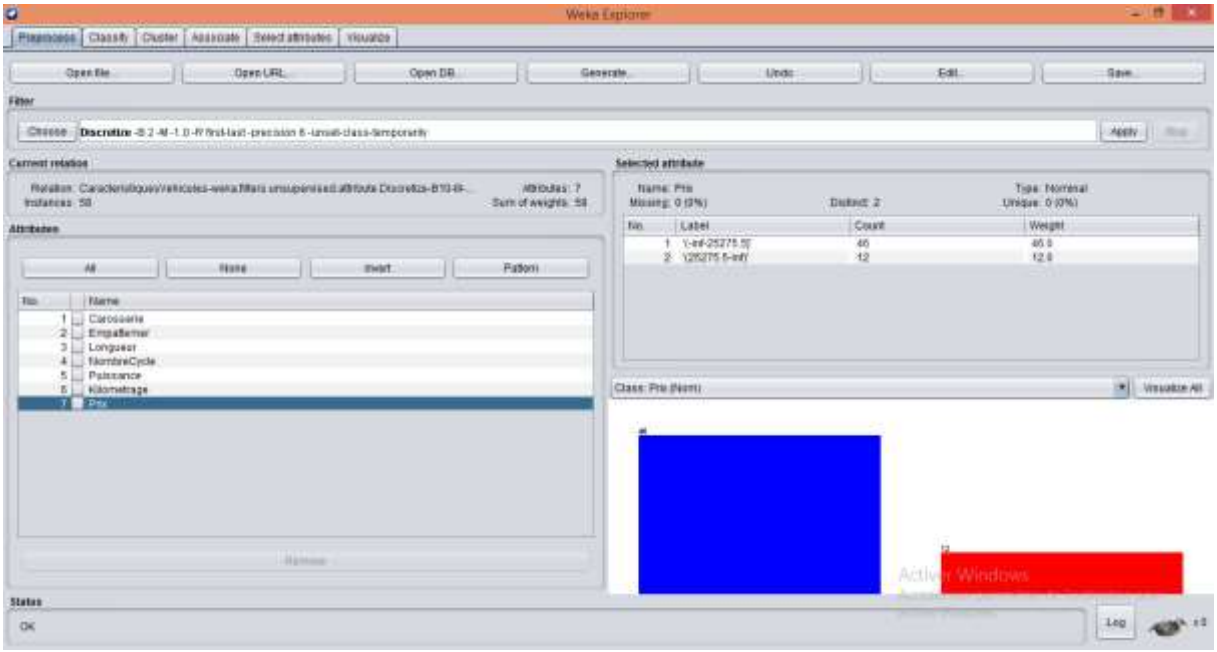


Figure 2:filtre discretize

Avec ce prétraitement on distingue deux classes ou cluster, bas (avec couleur rouge et un cout de 42) et élevé (avec couleur bleu avec un cout de 12) et chaque élément doit appartient à un seul cluster.

Question 3 :

Maintenant, on va effectuez un clustering avec la méthode K-means, nous utiliserons cet algorithme pour regrouper à partir des 6 attributs ou caractéristique pour caractériser les prix de véhicules résultants. L'avantage de cet algorithme est de gérer automatiquement un mélange d'attributs catégoriques et numériques. De plus, l'algorithme normalise automatiquement les attributs numériques lors des calculs de distance.

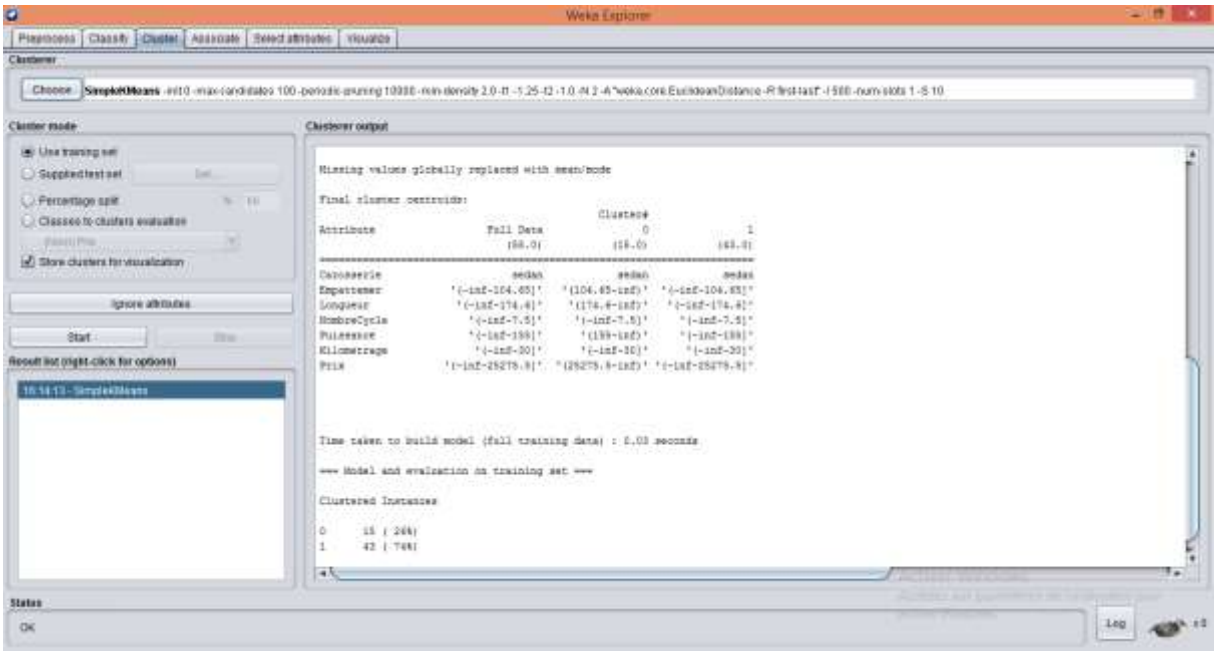


Figure 3: K-means

Apres le clustering on peut identifier le deux cluster 0 et 1 avec leurs instances 15 et 41.

Question 4 :

La différence entre le partitionnement obtenu au départ et le partitionnement obtenu avec la méthode k-means consiste au quelques éléments qui appartient au premier cluster devient appartient au deuxième cluster.

Question 5 :

En appliquant la méthode «elbow method » avec un intervalle de nombre de groupes k entre 2 et 20 puis on va schématiser un courbe en fonction de cout dans l'axe de y et le nombre de clusters dans l'axe de x. l'objectif de cette technique est pour obtenir le cluster optimale de cet ensemble des données.

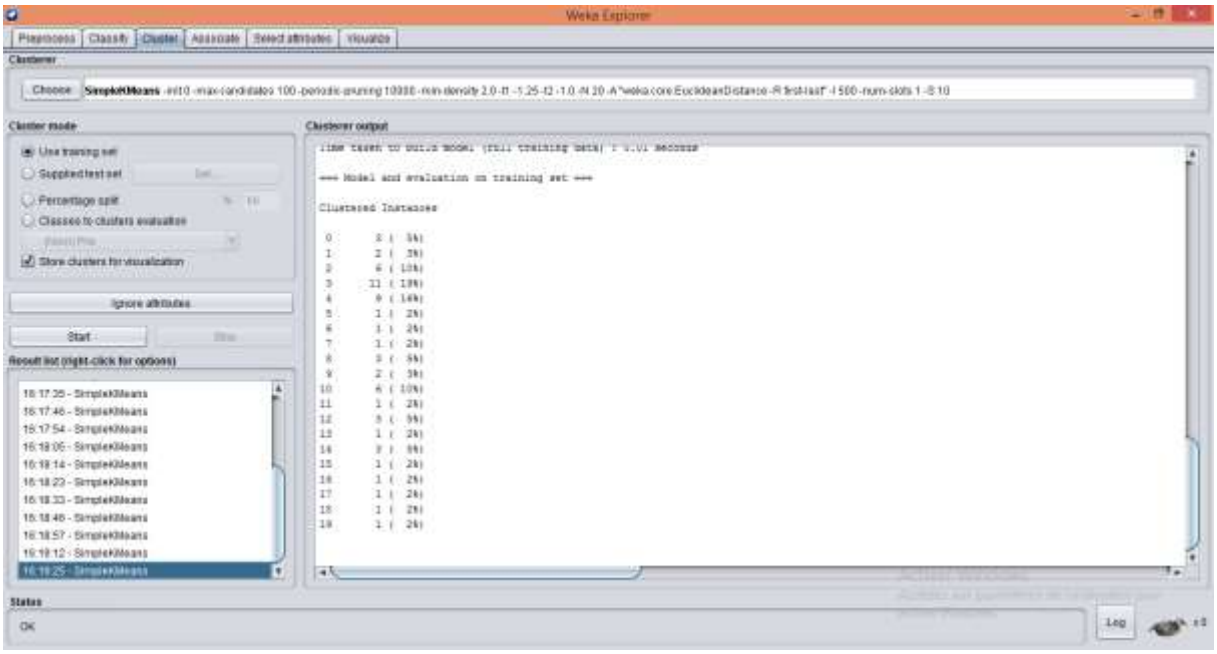


Figure 4: elbow method

La technique est très simple, chaque fois on va changer le nombre de cluster « incrémenter a 1 » et on restart l'algorithme pour obtenir le cout qui correspond à la sum of squared errors jusqu'à au nombre de cluster égal à 19. Ensuit on passe à l'étape de construction de schéma qui nous permet d'identifier le cluster optimale.

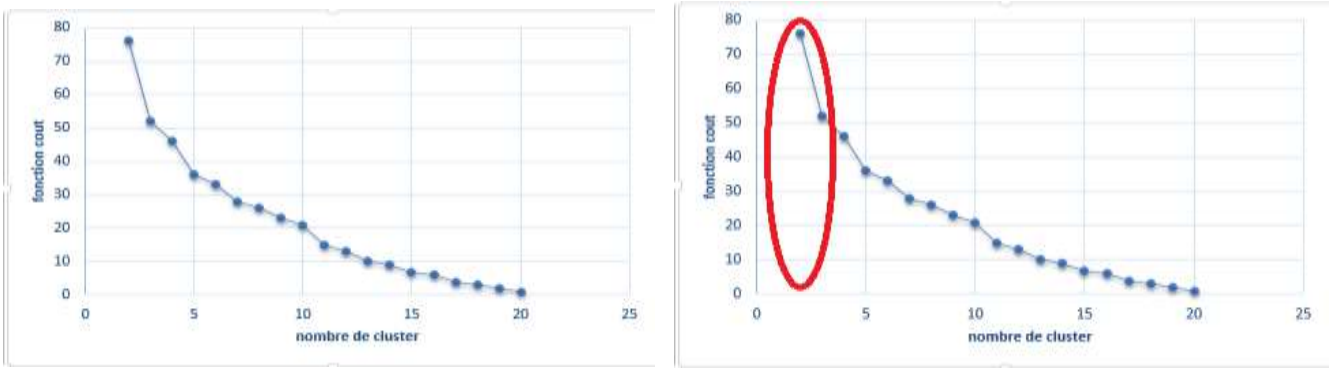


Figure 5: courbe « elbow method »

On peut conclure à partir de figure 5 que le cluster optimal est égale 3.

Question 6 :

Pour le cluster 3 la méthode K-means donne le résultat suivant

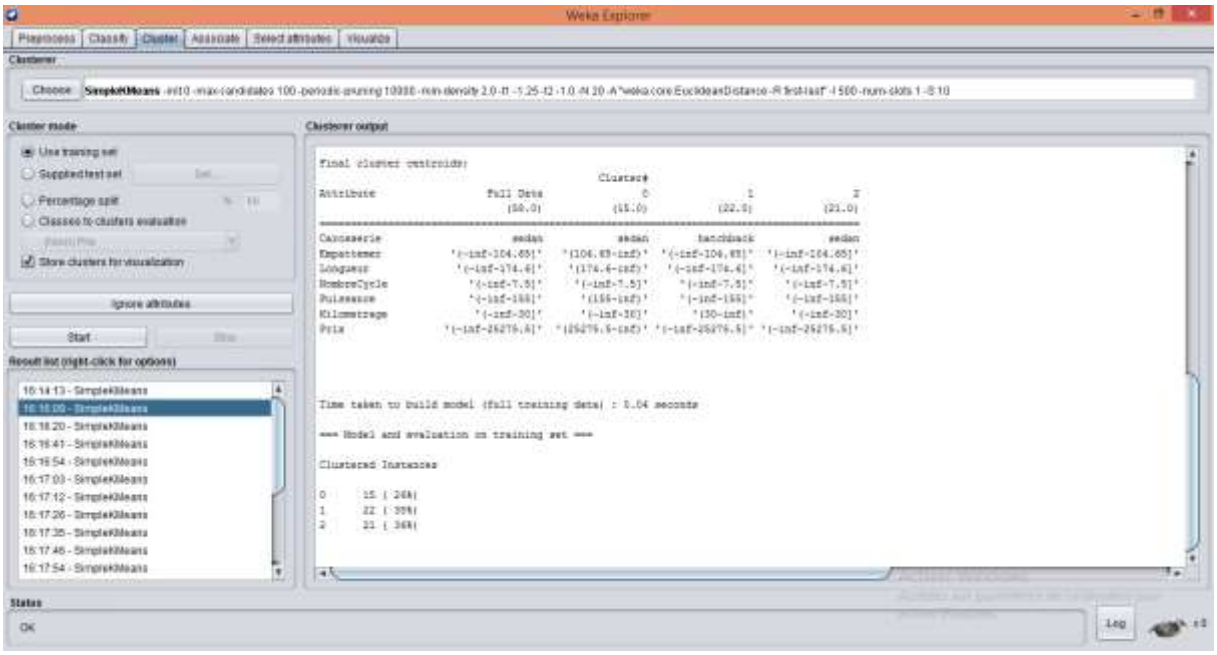


Figure 6: K-means de 3 cluster

La méthode k-means++ donne le résultat suivant :

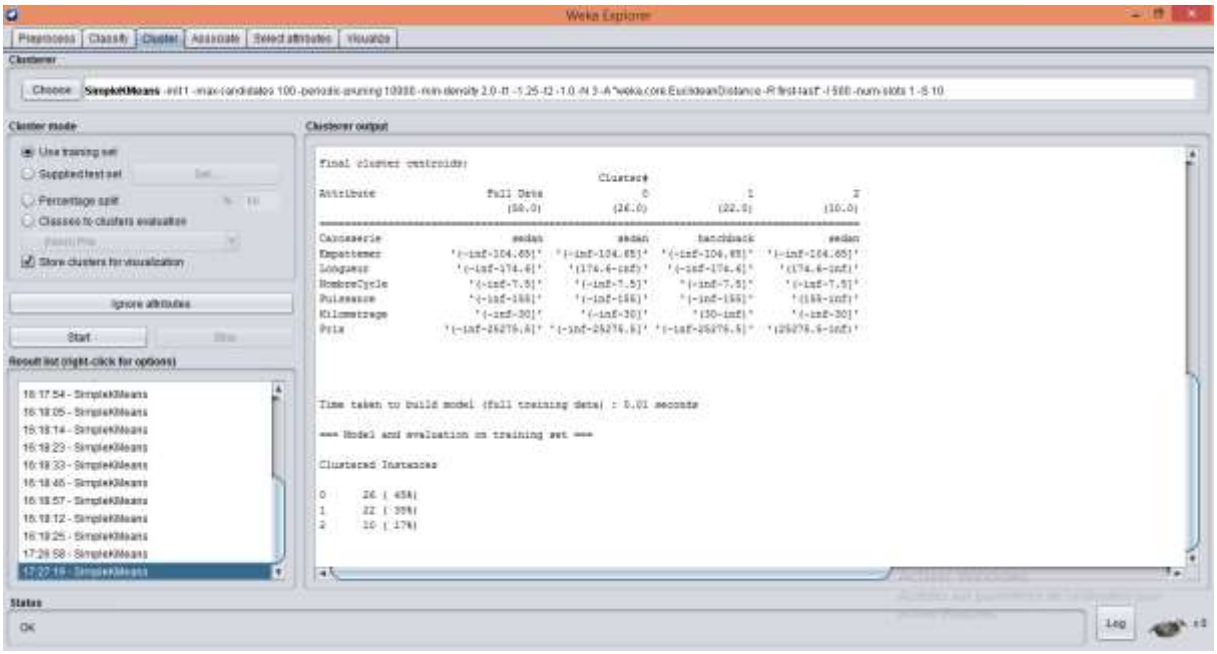


Figure 7: K-means++ de 3 cluster

On constate que La méthode k-means ++ est meilleur que k-means car le nombre des éléments dans le bloc .

Question 7 :

En fait le clustering avec les differents distances CAH .

En commence par la Distance la plus connu :

- distance euclidienne

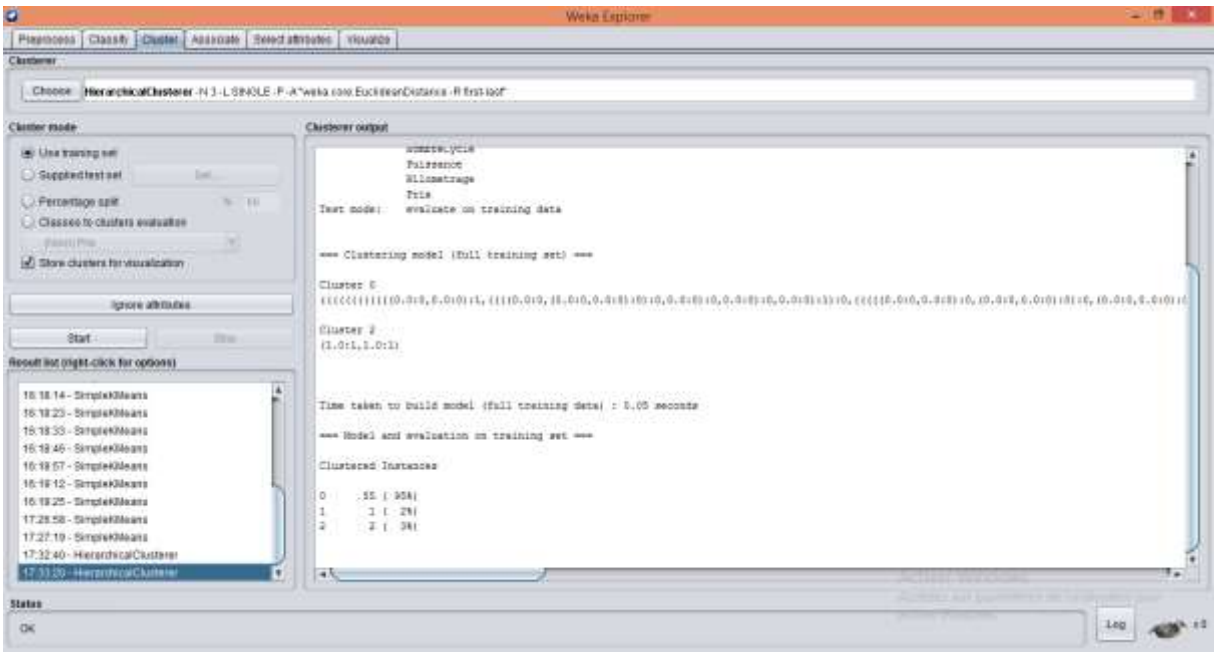


Figure 8: Distance euclidienne

La figure 8 montre que la résultat obtenu est très diffèrent que le résultat obtenu par K-means.

- Distance Manhattan :

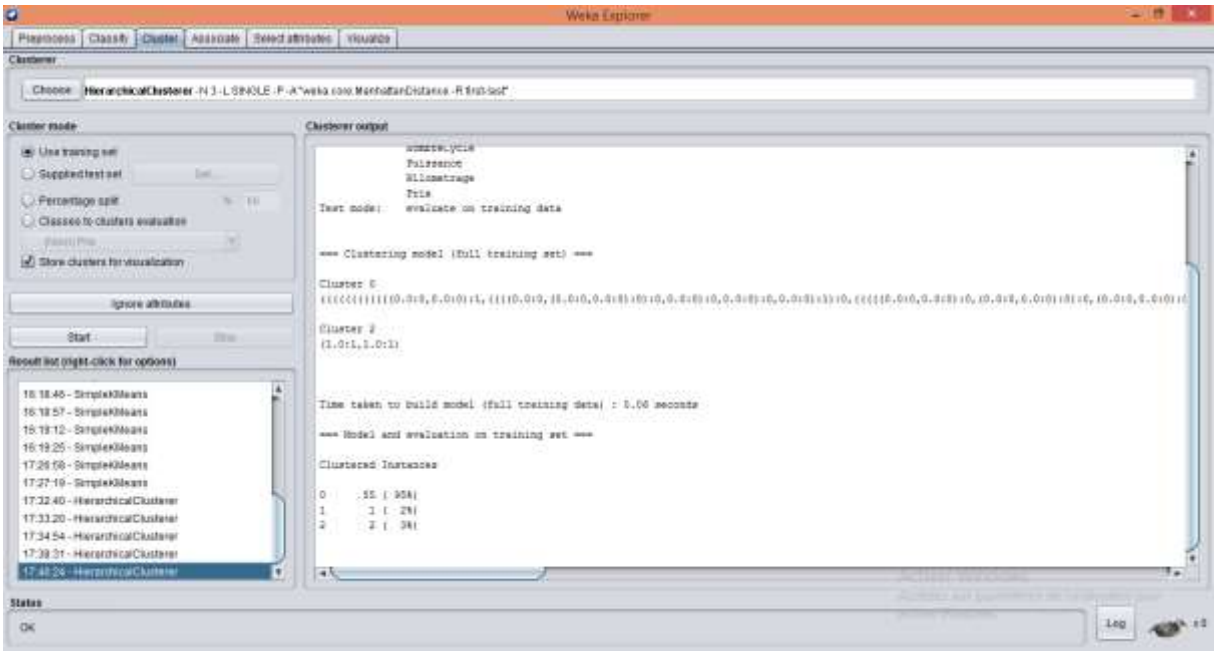


Figure 9: Distance Manhattan

Cette distance est similaire à la distance précédente donc elle est différente au K-means.

- Distance Minkowski :

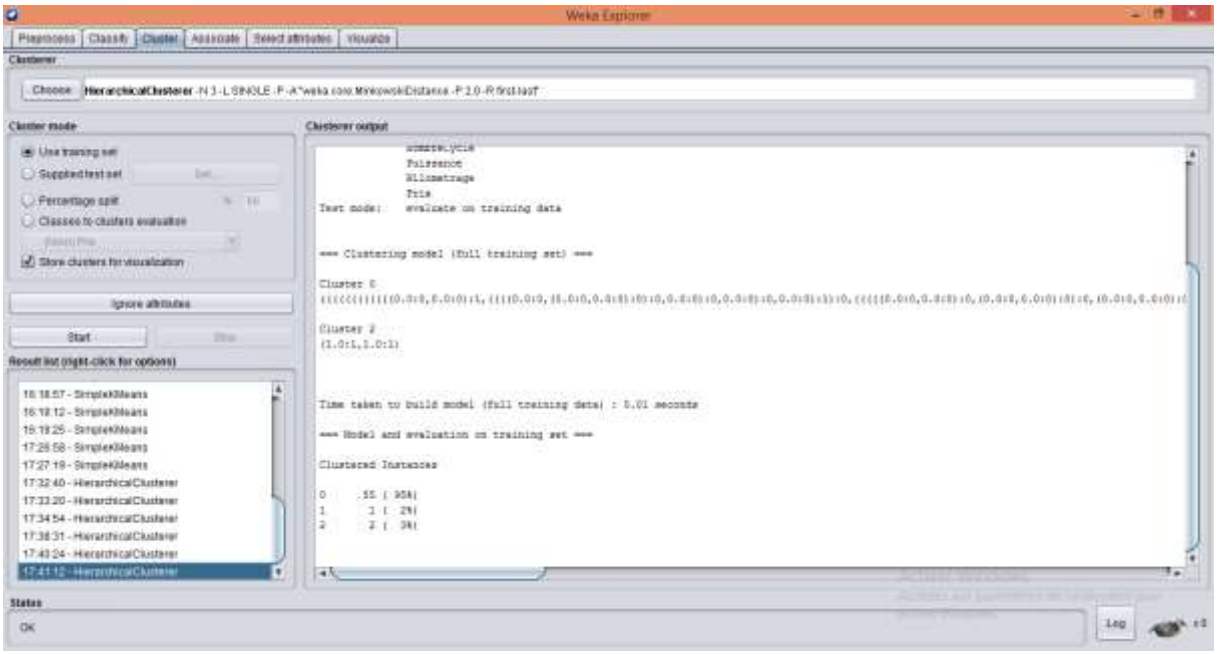


Figure 10: distance Minkowski

Même concept que la distance précédente.

Distance Chebychev :

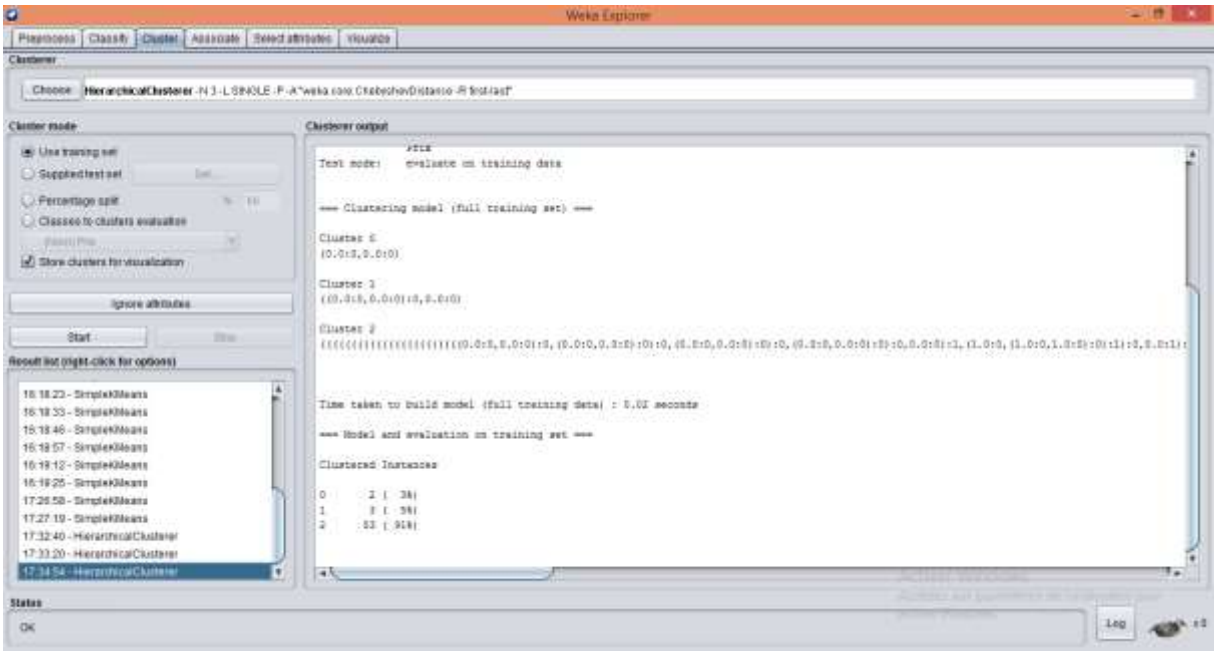


Figure 11: distance chebyshev

- Distance filtred distance :

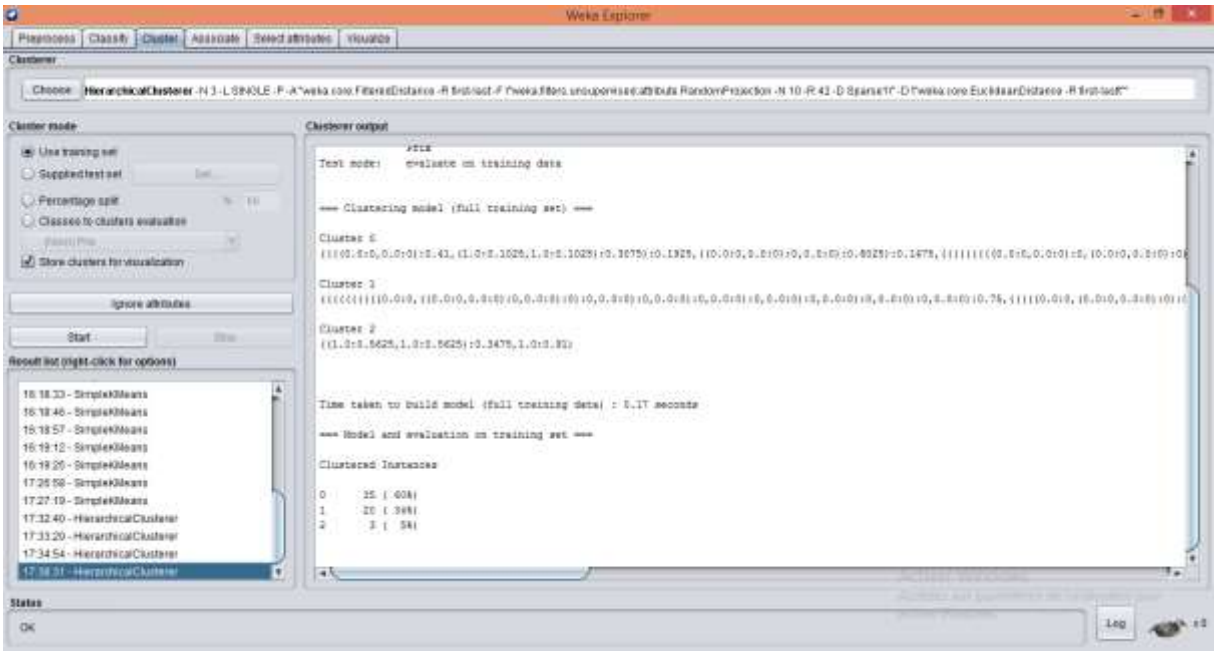


Figure 12: filtred distance

Cette distance est le meilleur car il est les plus approche au résultat de K-means.

Question 8 :

Pour améliorer les résultats de clustering je propose deux solutions :

- Augmenter les données d'entrées (fichier ARFF)
- Diminuer le nombre des attributs utilisés dans ce fichier.

Partie 2 : classification

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:

```

```

Je commence par l'importation de deux bibliothèques pandas et matplotlib.

In [2]:

```
file=pd.read_excel('fichiermodifier.xlsx')
```

Ensuite, je dois transformer les deux colonnes Carosserie et Nombre-cylindres du fichier CaractéristiqueVéhicules en chiffre : 1 pour Carosserie de type convertible 2 pour Carosserie de type hatchback 3 pour Carosserie de type sedan 4 pour Carosserie de type wagon 5 pour Carosserie de type hardtop

et aprés et après l'importation du ce dernier dans le Jupiter.

In [3]:

```
file.head()
```

Out[3]:

	Identifiant	Carosserie	Empattement	longueur	Nombre-cylindres	Puissance	KilometrageMoyen	Prix
0	0	1	88.6	168.8	4	111	21	13495
1	1	1	88.6	168.8	4	111	21	16500
2	2	2	94.5	171.2	6	154	19	16500
3	3	3	99.8	176.6	4	102	24	13950
4	4	3	99.4	176.6	5	115	18	17450

Maintenant, je choisi le deux variables X qui correspond au caractéristiques et la cible y qui correspond a la prix.

In [6]:

```
X = file.iloc[:,1:7]
y = file.iloc[:,-1]
```

In [7]:

```
X
```

Out[7]:

	Carosserie	Empattement	longueur	Nombre-cylindres	Puissance	KilometrageMoyen
0	1	88.6	168.8	4	111	21
1	1	88.6	168.8	4	111	21
2	2	94.5	171.2	6	154	19
3	3	99.8	176.6	4	102	24
4	3	99.4	176.6	5	115	18
5	3	99.8	177.3	5	110	19

	Carosserie	Empattement	longueur	Nombre-cylindres	Puissance	KilometrageMoyen
6	4	105.8	192.7	5	110	19
7	3	101.2	176.8	4	101	23
8	3	101.2	176.8	4	101	23
9	3	101.2	176.8	6	121	21
10	3	103.5	189.0	6	182	16
11	3	103.5	193.8	6	182	16
12	3	110.0	197.0	6	182	15
13	2	88.4	141.1	3	48	47
14	2	94.5	155.9	4	70	38
15	3	94.5	158.8	4	70	38
16	2	93.7	157.3	4	68	31
17	2	93.7	157.3	4	68	31
18	4	96.5	157.1	4	76	30
19	3	96.5	175.4	4	101	24
20	3	96.5	169.1	4	100	25
21	3	94.3	170.7	4	78	24
22	3	113.0	199.6	6	176	15
23	3	113.0	199.6	6	176	15
24	3	102.0	191.7	13	262	13
25	2	93.1	159.1	4	68	30
26	2	93.1	159.1	4	68	31
27	2	93.1	159.1	4	68	31
28	2	95.3	169.0	2	101	17
29	3	104.9	175.0	4	72	31
30	3	110.0	190.9	5	123	22
31	4	110.0	190.9	5	123	22
32	3	120.9	208.1	8	184	14
33	5	112.0	199.2	8	184	14
34	2	93.7	157.3	4	68	37
35	2	93.7	157.3	4	68	31
36	3	96.3	172.4	4	88	25
37	3	96.3	172.4	4	88	25
38	3	94.5	165.3	4	55	45
39	3	94.5	165.3	4	69	31
40	3	94.5	165.3	4	69	31
41	2	94.5	170.2	4	69	31
42	3	100.4	184.6	6	152	19
43	5	89.5	168.9	6	207	17
44	1	89.5	168.9	6	207	17
45	2	95.7	158.7	4	62	35
46	2	95.7	158.7	4	62	31
47	2	95.7	158.7	4	62	31
48	4	95.7	169.7	4	62	31
49	4	95.7	169.7	4	62	27
50	4	95.7	169.7	4	62	27

	Carosserie	Empattement	longueur	Nombre-cylindres	Puissance	KilometrageMoyen
51	4	104.5	187.8	6	156	19
52	3	97.3	171.7	4	52	37
53	3	97.3	171.7	4	85	27
54	3	97.3	171.7	4	52	37
55	3	97.3	171.7	4	100	26
56	3	104.3	188.8	4	114	23
57	4	104.3	188.8	4	114	23

In [8]:

```
y
```

Out[8]:

```
0     13495
1     16500
2     16500
3     13950
4     17450
5     15250
6     18920
7     16430
8     16925
9     20970
10    30760
11    41315
12    36880
13     5151
14     6295
15     6575
16     6377
17     6229
18     7295
19    12945
20    10345
21     6785
22    32250
23    35550
24    36000
25     5195
26     6095
27     6795
28    11845
29    18344
30    25552
31    28248
32    40960
33    45400
34     5389
35     6189
36     6989
37     8189
38     7099
39     6649
40     6849
41     7349
42    13499
43    34028
44    37028
45     5348
46     6338
47     6488
48     6918
49     7898
50     8778
51    15750
52     7775
53     7975
```

```
54     7995
55     9995
56    12940
57    13415
Name: Prix, dtype: int64
```

Comme montre les teste ci-dessus que l'importation du fichier et la création de deux variables X et y a été bien passée en commence le test des algorithmes.

SVM :

In [9]:

```
from sklearn import svm
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.5, random_stat
e=0)
v_clf = svm.SVC()
v_clf.fit(X_train1, y_train1)
```

Out[9]:

```
SVC()
```

In [10]:

```
v_clf.predict(X_test1)
```

Out[10]:

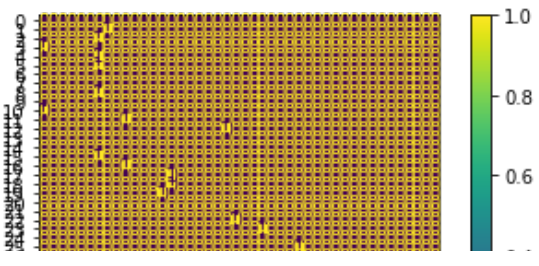
```
array([ 6377,  6377, 15750,  5195, 36880, 15750, 36880,  6488, 35550,
        8778, 20970, 36880,  6649,  6649, 35550,  6649,  6989, 16925,
        6377, 13415, 10345, 13415,  7099,  6377,  7099,  5195,  6377,
       13950, 13415], dtype=int64)
```

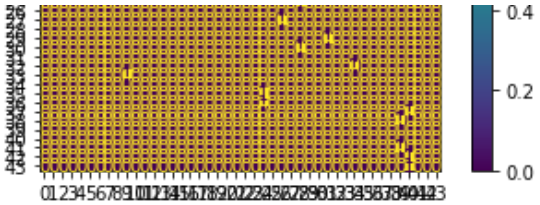
In [11]:

```
#Pour voit la différence entre les algorithmes je dois effectuer à chaque fois la matrice
de confusion et le score du chacun.
#Matrice de confusion de l'algorithme SVM
plot_confusion_matrix(v_clf, X_test1, y_test1)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-11-b92ccdae29fd> in <module>
      1 #Pour voit la différence entre les algorithmes je dois effectuer à chaque fois la
matrice de confusion et le score du chacun.
----> 2 plot_confusion_matrix(v_clf, X_test1, y_test1)
      3 plt.show()

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot_confusion_matrix(estimator, X, y_true, labels, sample_weight, normalize, display_labe
ls, include_values, xticks_rotation, values_format, cmap, ax)
    229     disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    230                                   display_labels=display_labels)
--> 231     return disp.plot(include_values=include_values,
    232                      cmap=cmap, ax=ax, xticks_rotation=xticks_rotation,
    233                      values_format=values_format)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
```

```
**kwargs)
     70                             FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot(self, include_values, cmap, xticks_rotation, values_format, ax)
    118
    119         fig.colorbar(self.im_, ax=ax)
--> 120         ax.set(xticks=np.arange(n_classes),
    121                yticks=np.arange(n_classes),
    122                xticklabels=display_labels,

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in set(self, **kwargs)
   1111         if move_color_to_start:
   1112             kwargs = {"color": kwargs.pop("color"), **kwargs}
-> 1113         return self.update(kwargs)
   1114
   1115     def findobj(self, match=None, include_self=True):

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in update(self, props)
    996                         raise AttributeError(f"{type(self).__name__!r} object "
    997                                              f"has no property {k!r}")
--> 998                     ret.append(func(v))
    999         if ret:
   1000             self.pchanged()

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in wrapper(self, *arg
s, **kwargs)
     61
     62         def wrapper(self, *args, **kwargs):
---> 63             return get_method(self)(*args, **kwargs)
     64
     65         wrapper.__module__ = owner.__module__

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py in wrapper(*ar
gs, **kwargs)
    449                 "parameter will become keyword-only %(removal)s.",
    450                 name=name, obj_type=f"parameter of {func.__name__}()")
--> 451         return func(*args, **kwargs)
    452
    453     return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in _set_ticklabels(self, la
bels, fontdict, minor, **kwargs)
   1791         if fontdict is not None:
   1792             kwargs.update(fontdict)
-> 1793         return self.set_ticklabels(labels, minor=minor, **kwargs)
   1794
   1795     @cbook._make_keyword_only("3.2", "minor")

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in set_ticklabels(self, tic
klabels, minor, **kwargs)
   1712             # remove all tick labels, so only error for > 0 ticklabels
   1713             if len(locator.locs) != len(ticklabels) and len(ticklabels) != 0:
-> 1714                 raise ValueError(
   1715                     "The number of FixedLocator locations"
   1716                     f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (44), usually from a call to set_ticks,
does not match the number of ticklabels (29).
```





On peut constater que plus la couleur tend vers la couleur jaune l'algorithme est plus performant.

In [13]:

```
v_clf.score(X_train1,y_train1)
#score de l'algorithme SVM
```

Out[13]:

```
0.8620689655172413
```

LogisticRegression :

In [14]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.5, random_stat
e=0)
lreg = LogisticRegression().fit(X_train2, y_train2)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: Converg
enceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [15]:

```
lreg.predict(X_test2)
```

Out[15]:

```
array([ 6377,  6229, 15750,  5195, 36880, 15750, 36880,  6488, 35550,
        8778, 20970, 36880,  6338,  6649, 35550,  6649,  6989, 16925,
        6377, 20970, 18920, 13415,  7099,  6377,  7099,  5195,  6229,
       13415, 20970], dtype=int64)
```

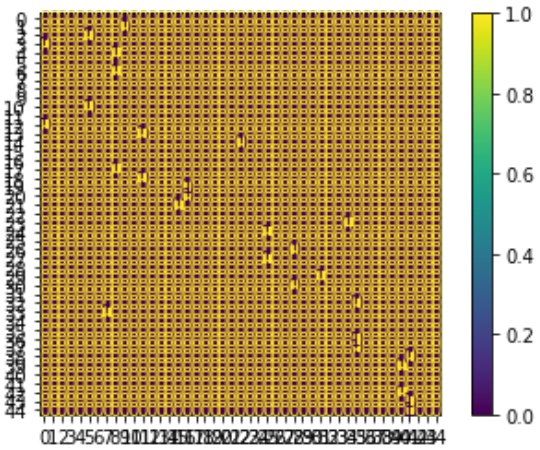
In [16]:

```
plot_confusion_matrix(lreg, X_test2, y_test2)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-30e818939c7a> in <module>
----> 1 plot_confusion_matrix(lreg, X_test2, y_test2)
      2 plt.show()

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot_confusion_matrix(estimator, X, y_true, labels, sample_weight, normalize, display_labe
ls, include_values, xticks_rotation, values_format, cmap, ax)
    229     disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    230                                   display_labels=display_labels)
--> 231     return disp.plot(include_values=include_values,
    232                      cmap=cmap, ax=ax, xticks_rotation=xticks_rotation,
    233                      values_format=values_format)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot(self, include_values, cmap, xticks_rotation, values_format, ax)
    118
    119         fig.colorbar(self.im_, ax=ax)
--> 120         ax.set(xticks=np.arange(n_classes),
    121                yticks=np.arange(n_classes),
    122                xticklabels=display_labels,

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in set(self, **kwargs)
   1111         if move_color_to_start:
   1112             kwargs = {"color": kwargs.pop("color"), **kwargs}
-> 1113         return self.update(kwargs)
   1114
   1115     def findobj(self, match=None, include_self=True):

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in update(self, props)
    996                         raise AttributeError(f"{type(self).__name__!r} object "
    997                                              f"has no property {k!r}")
--> 998                     ret.append(func(v))
    999         if ret:
   1000             self.pchanged()

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in wrapper(self, *arg
s, **kwargs)
     61
     62         def wrapper(self, *args, **kwargs):
---> 63             return get_method(self)(*args, **kwargs)
     64
     65         wrapper.__module__ = owner.__module__

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py in wrapper(*ar
gs, **kwargs)
    449                 "parameter will become keyword-only %(removal)s.",
    450                 name=name, obj_type=f"parameter of {func.__name__}()")
--> 451         return func(*args, **kwargs)
    452
    453     return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in _set_ticklabels(self, la
bels, fontdict, minor, **kwargs)
   1791         if fontdict is not None:
   1792             kwargs.update(fontdict)
-> 1793         return self.set_ticklabels(labels, minor=minor, **kwargs)
   1794
   1795     @cbook._make_keyword_only("3.2", "minor")

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in set_ticklabels(self, tic
klabels, minor, **kwargs)
   1712             # remove all tick labels, so only error for > 0 ticklabels
   1713             if len(locator.locs) != len(ticklabels) and len(ticklabels) != 0:
-> 1714                 raise ValueError(
   1715                     "The number of FixedLocator locations"
   1716                     f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (45), usually from a call to set_ticks,
does not match the number of ticklabels (29).
```

does not match the number of ticklabels (29).



In [17]:

```
lreg.score(X_train2,y_train2)
```

Out[17]:

```
0.8620689655172413
```

Naïve bayes :

In [18]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split

X_train3, X_test3, y_train3, y_test3 = train_test_split(X, y, test_size=0.5, random_stat
e=0)
nvb = GaussianNB().fit(X_train3,y_train3)
```

In [19]:

```
nvb.score(X_train3,y_train3)
```

Out[19]:

```
0.8620689655172413
```

In [20]:

```
nvb.predict(X_test3)
```

Out[20]:

```
array([ 6229,  6229, 15750,  5195, 36880, 15750, 36880,  6338, 35550,
        7898, 20970, 36880,  6649,  6649, 35550,  6649,  6989, 16925,
        6229, 13415, 10345, 13415,  7099,  6229,  7099,  5195,  6229,
       13950, 13415], dtype=int64)
```

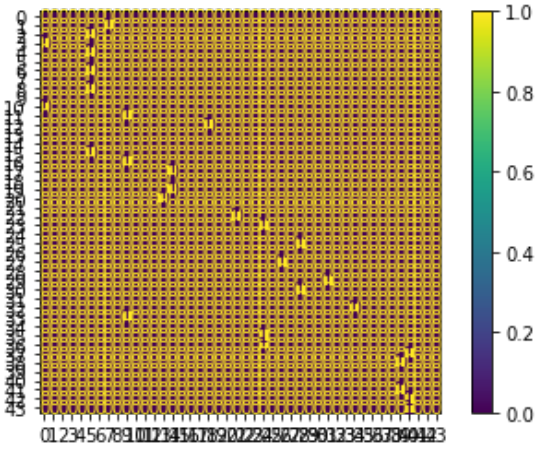
In [21]:

```
plot_confusion_matrix(nvb, X_test3, y_test3)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-21-1bfe02f43cc6> in <module>
----> 1 plot_confusion_matrix(nvb, X_test3, y_test3)
      2 plt.show()

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
```

```
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot_confusion_matrix(estimator, X, y_true, labels, sample_weight, normalize, display_labe
ls, include_values, xticks_rotation, values_format, cmap, ax)
    229     disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    230                                   display_labels=display_labels)
--> 231     return disp.plot(include_values=include_values,
    232                      cmap=cmap, ax=ax, xticks_rotation=xticks_rotation,
    233                      values_format=values_format)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot(self, include_values, cmap, xticks_rotation, values_format, ax)
    118
    119         fig.colorbar(self.im_, ax=ax)
--> 120         ax.set(xticks=np.arange(n_classes),
    121                yticks=np.arange(n_classes),
    122                xticklabels=display_labels,

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in set(self, **kwargs)
   1111         if move_color_to_start:
   1112             kwargs = {"color": kwargs.pop("color"), **kwargs}
-> 1113         return self.update(kwargs)
   1114
   1115     def findobj(self, match=None, include_self=True):

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in update(self, props)
    996                         raise AttributeError(f"{type(self).__name__!r} object "
    997                                              f"has no property {k!r}")
--> 998                     ret.append(func(v))
    999         if ret:
   1000             self.pchanged()

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in wrapper(self, *arg
s, **kwargs)
     61
     62         def wrapper(self, *args, **kwargs):
---> 63             return get_method(self)(*args, **kwargs)
     64
     65         wrapper.__module__ = owner.__module__

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py in wrapper(*ar
gs, **kwargs)
    449                 "parameter will become keyword-only %(removal)s.",
    450                 name=name, obj_type=f"parameter of {func.__name__}()")
--> 451         return func(*args, **kwargs)
    452
    453     return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in _set_ticklabels(self, la
bels, fontdict, minor, **kwargs)
   1791         if fontdict is not None:
   1792             kwargs.update(fontdict)
-> 1793         return self.set_ticklabels(labels, minor=minor, **kwargs)
   1794
   1795     @cbook._make_keyword_only("3.2", "minor")

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in set_ticklabels(self, tic
klabels, minor, **kwargs)
   1712             # remove all tick labels, so only error for > 0 ticklabels
   1713             if len(locator.locs) != len(ticklabels) and len(ticklabels) != 0:
-> 1714                 raise ValueError(
   1715                     "The number of FixedLocator locations"
```

```
1716                    f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (44), usually from a call to set_ticks,
does not match the number of ticklabels (29).
```



In [23]:

```
file.iloc[44,:]
```

Out[23]:

```
Identifiant            62.0
Carosserie              1.0
Empattement            89.5
longueur              168.9
Nombre-cylindres        6.0
Puissance             207.0
KilometrageMoyen       17.0
Prix                37028.0
Name: 44, dtype: float64
```

KNN :

In [24]:

```
from sklearn.neighbors import KNeighborsClassifier

X_train3, X_test3, y_train3, y_test3 = train_test_split(X, y, test_size=0.5, random_stat
e=0)
knn = KNeighborsClassifier(n_neighbors = 1).fit(X_train3,y_train3)
```

In [25]:

```
knn.score(X_train3,y_train3)
```

Out[25]:

```
0.8620689655172413
```

In [26]:

```
knn.predict(X_test3)
```

Out[26]:

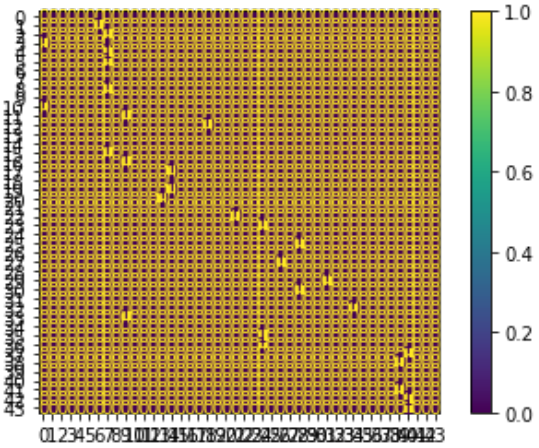
```
array([ 6377,  6377, 15750,  5195, 36880, 15750, 36880,  6338, 35550,
        7898, 20970, 36880,  6649,  6649, 35550,  6649,  6989, 16925,
        6377, 13415, 10345, 13415,  7099,  6377,  7099,  5195,  6377,
       13950, 13415], dtype=int64)
```

In [27]:

```
plot_confusion_matrix(knn, X_test3, y_test3)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-27-70dbb7b96b1a> in <module>
----> 1 plot_confusion_matrix(knn, X_test3, y_test3)
      2 plt.show()

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot_confusion_matrix(estimator, X, y_true, labels, sample_weight, normalize, display_labe
ls, include_values, xticks_rotation, values_format, cmap, ax)
    229     disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    230                                   display_labels=display_labels)
--> 231     return disp.plot(include_values=include_values,
    232                      cmap=cmap, ax=ax, xticks_rotation=xticks_rotation,
    233                      values_format=values_format)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot(self, include_values, cmap, xticks_rotation, values_format, ax)
    118
    119         fig.colorbar(self.im_, ax=ax)
--> 120         ax.set(xticks=np.arange(n_classes),
    121                yticks=np.arange(n_classes),
    122                xticklabels=display_labels,

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in set(self, **kwargs)
   1111         if move_color_to_start:
   1112             kwargs = {"color": kwargs.pop("color"), **kwargs}
-> 1113         return self.update(kwargs)
   1114
   1115     def findobj(self, match=None, include_self=True):

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in update(self, props)
    996                         raise AttributeError(f"{type(self).__name__!r} object "
    997                                              f"has no property {k!r}")
--> 998                     ret.append(func(v))
    999         if ret:
   1000             self.pchanged()

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in wrapper(self, *arg
s, **kwargs)
     61
     62         def wrapper(self, *args, **kwargs):
---> 63             return get_method(self)(*args, **kwargs)
     64
     65         wrapper.__module__ = owner.__module__

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py in wrapper(*ar
gs, **kwargs)
    449                 "parameter will become keyword-only %(removal)s.",
    450                 name=name, obj_type=f"parameter of {func.__name__}()")
--> 451         return func(*args, **kwargs)
    452
    453     return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in _set_ticklabels(self, la
bels, fontdict, minor, **kwargs)
   1791         if fontdict is not None:
```

```
   1792             kwargs.update(fontdict)
-> 1793         return self.set_ticklabels(labels, minor=minor, **kwargs)
   1794
   1795     @cbook._make_keyword_only("3.2", "minor")

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in set_ticklabels(self, tic
klabels, minor, **kwargs)
   1712             # remove all tick labels, so only error for > 0 ticklabels
   1713             if len(locator.locs) != len(ticklabels) and len(ticklabels) != 0:
-> 1714                 raise ValueError(
   1715                     "The number of FixedLocator locations"
   1716                     f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (44), usually from a call to set_ticks,
does not match the number of ticklabels (29).
```



Arbres de décision :

In [28]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
X_train4, X_test4, y_train4, y_test4 = train_test_split(X, y, test_size=0.5, random_stat
e=0)
ard = DecisionTreeClassifier(random_state=0).fit(X_train4,y_train4)
```

In [29]:

```
ard.score(X_train4,y_train4)
```

Out[29]:

```
0.8620689655172413
```

In [30]:

```
ard.predict(X_test4)
```

Out[30]:

```
array([ 6229,  6229, 20970,  5195, 36880, 20970, 36000,  6338, 35550,
        7898, 13950, 36880,  7975,  6649, 36000,  6785,  6989, 16925,
        6229, 18920, 10345, 16925,  7099,  6229,  7099,  5195,  5195,
       13950, 16925], dtype=int64)
```

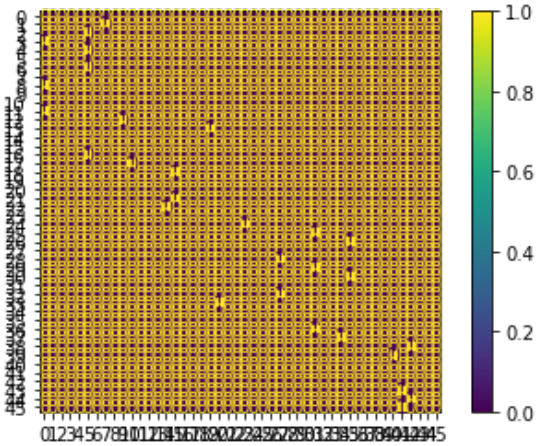
In [31]:

```
plot_confusion_matrix(ard, X_test4, y_test4)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-31-93a975c7560e> in <module>
----> 1 plot_confusion_matrix(ard, X_test4, y_test4)
      2 plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot_confusion_matrix(estimator, X, y_true, labels, sample_weight, normalize, display_labe
ls, include_values, xticks_rotation, values_format, cmap, ax)
    229     disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    230                                   display_labels=display_labels)
--> 231     return disp.plot(include_values=include_values,
    232                      cmap=cmap, ax=ax, xticks_rotation=xticks_rotation,
    233                      values_format=values_format)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot(self, include_values, cmap, xticks_rotation, values_format, ax)
    118
    119         fig.colorbar(self.im_, ax=ax)
--> 120         ax.set(xticks=np.arange(n_classes),
    121                yticks=np.arange(n_classes),
    122                xticklabels=display_labels,

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in set(self, **kwargs)
   1111         if move_color_to_start:
   1112             kwargs = {"color": kwargs.pop("color"), **kwargs}
-> 1113         return self.update(kwargs)
   1114
   1115     def findobj(self, match=None, include_self=True):

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in update(self, props)
    996                         raise AttributeError(f"{type(self).__name__!r} object "
    997                                              f"has no property {k!r}")
--> 998                     ret.append(func(v))
    999         if ret:
   1000             self.pchanged()

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in wrapper(self, *arg
s, **kwargs)
     61
     62         def wrapper(self, *args, **kwargs):
---> 63             return get_method(self)(*args, **kwargs)
     64
     65         wrapper.__module__ = owner.__module__

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py in wrapper(*ar
gs, **kwargs)
    449                 "parameter will become keyword-only %(removal)s.",
    450                 name=name, obj_type=f"parameter of {func.__name__}()")
--> 451         return func(*args, **kwargs)
    452
    453     return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in _set_ticklabels(self, la
bels, fontdict, minor, **kwargs)
   1791         if fontdict is not None:
   1792             kwargs.update(fontdict)
-> 1793         return self.set_ticklabels(labels, minor=minor, **kwargs)
   1794
   1795     @cbook._make_keyword_only("3.2", "minor")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in set_ticklabels(self, tic
klabels, minor, **kwargs)
   1712             # remove all tick labels, so only error for > 0 ticklabels
   1713             if len(locator.locs) != len(ticklabels) and len(ticklabels) != 0:
-> 1714                 raise ValueError(
   1715                     "The number of FixedLocator locations"
   1716                     f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (46), usually from a call to set_ticks,
does not match the number of ticklabels (29).
```



Multilayer Perceptron :

In [32]:

```
from sklearn.neural_network import MLPClassifier
X_train5, X_test5, y_train5, y_test5 = train_test_split(X, y, test_size=0.5, random_stat
e=0)
mlp = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=
1).fit(X_train5,y_train5)
```

In [33]:

```
mlp.score(X_train5, y_train5)
```

Out[33]:

```
0.06896551724137931
```

In [34]:

```
mlp.predict(X_test5)
```

Out[34]:

```
array([34028,  7099,  6989, 34028,  6989,  6989,  6989,  7099,  6989,
       34028,  6989,  6989, 34028, 34028,  6989, 34028,  6989,  6989,
        7099,  6989,  6989,  6989, 34028, 34028, 34028, 34028,  7099,
        6989,  6989], dtype=int64)
```

In [35]:

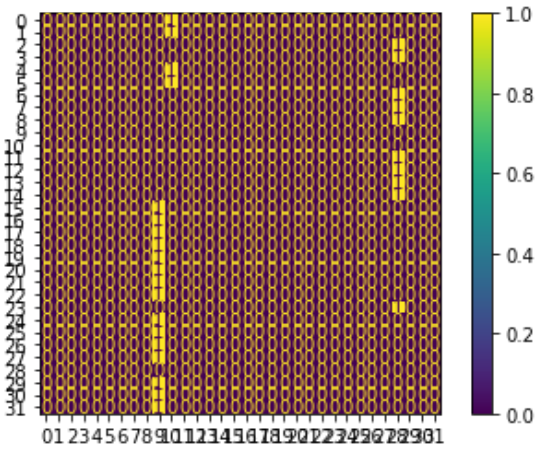
```
plot_confusion_matrix(mlp, X_test5, y_test5)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-35-1b96033ac089> in <module>
----> 1 plot_confusion_matrix(mlp, X_test5, y_test5)
      2 plt.show()

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
```

```
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot_confusion_matrix(estimator, X, y_true, labels, sample_weight, normalize, display_labe
ls, include_values, xticks_rotation, values_format, cmap, ax)
    229     disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    230                                   display_labels=display_labels)
--> 231     return disp.plot(include_values=include_values,
    232                      cmap=cmap, ax=ax, xticks_rotation=xticks_rotation,
    233                      values_format=values_format)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot(self, include_values, cmap, xticks_rotation, values_format, ax)
    118
    119         fig.colorbar(self.im_, ax=ax)
--> 120         ax.set(xticks=np.arange(n_classes),
    121                yticks=np.arange(n_classes),
    122                xticklabels=display_labels,

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in set(self, **kwargs)
   1111         if move_color_to_start:
   1112             kwargs = {"color": kwargs.pop("color"), **kwargs}
-> 1113         return self.update(kwargs)
   1114
   1115     def findobj(self, match=None, include_self=True):

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in update(self, props)
    996                         raise AttributeError(f"{type(self).__name__!r} object "
    997                                              f"has no property {k!r}")
--> 998                     ret.append(func(v))
    999         if ret:
   1000             self.pchanged()

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in wrapper(self, *arg
s, **kwargs)
     61
     62         def wrapper(self, *args, **kwargs):
---> 63             return get_method(self)(*args, **kwargs)
     64
     65         wrapper.__module__ = owner.__module__

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py in wrapper(*ar
gs, **kwargs)
    449                 "parameter will become keyword-only %(removal)s.",
    450                 name=name, obj_type=f"parameter of {func.__name__}()")
--> 451         return func(*args, **kwargs)
    452
    453     return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in _set_ticklabels(self, la
bels, fontdict, minor, **kwargs)
   1791         if fontdict is not None:
   1792             kwargs.update(fontdict)
-> 1793         return self.set_ticklabels(labels, minor=minor, **kwargs)
   1794
   1795     @cbook._make_keyword_only("3.2", "minor")

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in set_ticklabels(self, tic
klabels, minor, **kwargs)
   1712             # remove all tick labels, so only error for > 0 ticklabels
   1713             if len(locator.locs) != len(ticklabels) and len(ticklabels) != 0:
-> 1714                 raise ValueError(
```

```
1715                     "The number of FixedLocator locations"
1716                     f" ({len(locator.locs)}), usually from a call to"
```

```
ValueError: The number of FixedLocator locations (32), usually from a call to set_ticks,
does not match the number of ticklabels (29).
```



Adaboost :

In [36]:

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
X_train6, X_test6, y_train6, y_test6 = train_test_split(X, y, test_size=0.5, random_stat
e=0)
ads = AdaBoostClassifier(n_estimators=100, random_state=0).fit(X_train6, y_train6)
```

In [37]:

```
ads.score(X_train6, y_train6)
```

Out[37]:

```
0.7241379310344828
```

In [38]:

```
ads.predict(X_test6)
```

Out[38]:

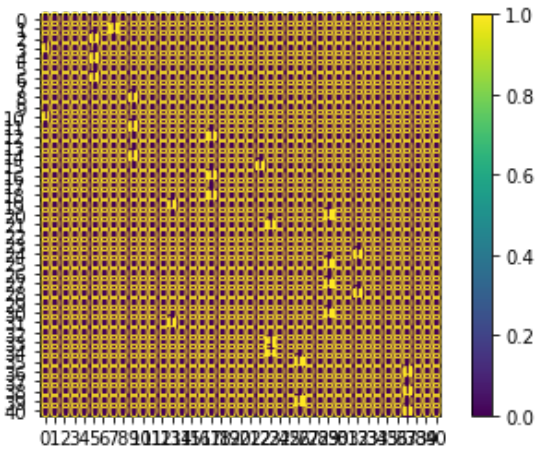
```
array([ 6229,  6229, 20970,  5195, 15750, 20970, 35550,  6338, 35550,
        7898, 16925, 15750,  6989,  6649, 35550, 12945,  6989, 16925,
        6229, 13415, 16925, 13415,  7898,  6649,  7898,  5195,  6649,
       16925, 13415], dtype=int64)
```

In [39]:

```
plot_confusion_matrix(ads, X_test6, y_test6)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-39-bd23ea934147> in <module>
----> 1 plot_confusion_matrix(ads, X_test6, y_test6)
      2 plt.show()

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
```

```
lot_confusion_matrix(estimator, X, y_true, labels, sample_weight, normalize, display_labe
ls, include_values, xticks_rotation, values_format, cmap, ax)
    229     disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    230                                   display_labels=display_labels)
--> 231     return disp.plot(include_values=include_values,
    232                      cmap=cmap, ax=ax, xticks_rotation=xticks_rotation,
    233                      values_format=values_format)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot(self, include_values, cmap, xticks_rotation, values_format, ax)
    118
    119         fig.colorbar(self.im_, ax=ax)
--> 120         ax.set(xticks=np.arange(n_classes),
    121                yticks=np.arange(n_classes),
    122                xticklabels=display_labels,

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in set(self, **kwargs)
   1111         if move_color_to_start:
   1112             kwargs = {"color": kwargs.pop("color"), **kwargs}
-> 1113         return self.update(kwargs)
   1114
   1115     def findobj(self, match=None, include_self=True):

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in update(self, props)
    996                         raise AttributeError(f"{type(self).__name__!r} object "
    997                                              f"has no property {k!r}")
--> 998                     ret.append(func(v))
    999         if ret:
   1000             self.pchanged()

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in wrapper(self, *arg
s, **kwargs)
     61
     62         def wrapper(self, *args, **kwargs):
---> 63             return get_method(self)(*args, **kwargs)
     64
     65         wrapper.__module__ = owner.__module__

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py in wrapper(*ar
gs, **kwargs)
    449                 "parameter will become keyword-only %(removal)s.",
    450                 name=name, obj_type=f"parameter of {func.__name__}()")
--> 451         return func(*args, **kwargs)
    452
    453     return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in _set_ticklabels(self, la
bels, fontdict, minor, **kwargs)
   1791         if fontdict is not None:
   1792             kwargs.update(fontdict)
-> 1793         return self.set_ticklabels(labels, minor=minor, **kwargs)
   1794
   1795     @cbook._make_keyword_only("3.2", "minor")

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in set_ticklabels(self, tic
klabels, minor, **kwargs)
   1712             # remove all tick labels, so only error for > 0 ticklabels
   1713             if len(locator.locs) != len(ticklabels) and len(ticklabels) != 0:
-> 1714                 raise ValueError(
   1715                     "The number of FixedLocator locations"
   1716                     f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (41), usually from a call to set_ticks,
does not match the number of ticklabels (29).
```



Random Forest :

In [40]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
X_train7, X_test7, y_train7, y_test7 = train_test_split(X, y, test_size=0.5, random_stat
e=0)
rf = RandomForestClassifier(max_depth=2, random_state=0).fit(X_train7,y_train7)
```

In [41]:

```
rf.score(X_train7,y_train7)
```

Out[41]:

```
0.8275862068965517
```

In [43]:

```
rf.predict(X_test7)
```

Out[43]:

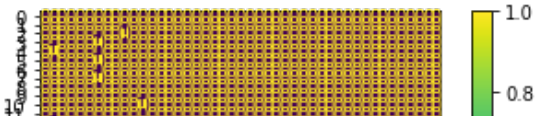
```
array([ 6229,  6229, 20970,  5195, 35550, 20970, 35550,  6488, 35550,
        8778, 12945, 20970,  6649,  6649, 35550,  6649,  6989, 16925,
        6229, 18920,  5151, 13415,  7099,  6377,  7099,  5195,  6649,
       12945, 36880], dtype=int64)
```

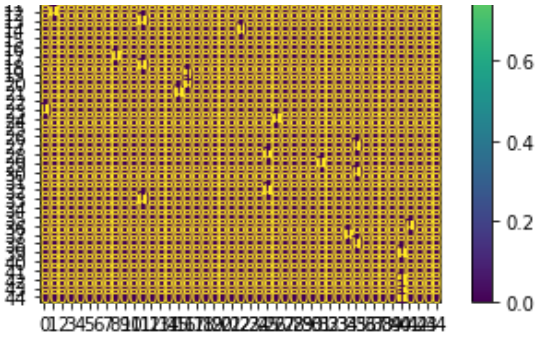
In [44]:

```
plot_confusion_matrix(rf, X_test7, y_test7)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-44-29fb098eae73> in <module>
----> 1 plot_confusion_matrix(rf, X_test7, y_test7)
      2 plt.show()

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot_confusion_matrix(estimator, X, y_true, labels, sample_weight, normalize, display_labe
ls, include_values, xticks_rotation, values_format, cmap, ax)
    229     disp = ConfusionMatrixDisplay(confusion_matrix=cm,
    230                                   display_labels=display_labels)
--> 231     return disp.plot(include_values=include_values,
```

```
    232                     cmap=cmap, ax=ax, xticks_rotation=xticks_rotation,
    233                     values_format=values_format)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args,
**kwargs)
     70                             FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 72         return f(**kwargs)
     73     return inner_f
     74

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py in p
lot(self, include_values, cmap, xticks_rotation, values_format, ax)
    118
    119         fig.colorbar(self.im_, ax=ax)
--> 120         ax.set(xticks=np.arange(n_classes),
    121                yticks=np.arange(n_classes),
    122                xticklabels=display_labels,

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in set(self, **kwargs)
   1111         if move_color_to_start:
   1112             kwargs = {"color": kwargs.pop("color"), **kwargs}
-> 1113         return self.update(kwargs)
   1114
   1115     def findobj(self, match=None, include_self=True):

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\artist.py in update(self, props)
    996                         raise AttributeError(f"{type(self).__name__!r} object "
    997                                              f"has no property {k!r}")
--> 998                 ret.append(func(v))
    999         if ret:
   1000             self.pchanged()

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in wrapper(self, *arg
s, **kwargs)
     61
     62         def wrapper(self, *args, **kwargs):
---> 63             return get_method(self)(*args, **kwargs)
     64
     65         wrapper.__module__ = owner.__module__

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py in wrapper(*ar
gs, **kwargs)
    449                 "parameter will become keyword-only %(removal)s.",
    450                 name=name, obj_type=f"parameter of {func.__name__}()")
--> 451         return func(*args, **kwargs)
    452
    453     return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in _set_ticklabels(self, la
bels, fontdict, minor, **kwargs)
   1791         if fontdict is not None:
   1792             kwargs.update(fontdict)
-> 1793         return self.set_ticklabels(labels, minor=minor, **kwargs)
   1794
   1795     @cbook._make_keyword_only("3.2", "minor")

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in set_ticklabels(self, tic
klabels, minor, **kwargs)
   1712             # remove all tick labels, so only error for > 0 ticklabels
   1713             if len(locator.locs) != len(ticklabels) and len(ticklabels) != 0:
-> 1714                 raise ValueError(
   1715                     "The number of FixedLocator locations"
   1716                     f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (45), usually from a call to set_ticks,
does not match the number of ticklabels (29).
```





finalement, selon les résultats (score) obtenu sur chaque algorithmes : *SVM : 0.8620689655172413*régression logistique : 0.8620689655172413 *Naïve bayes : 0.8620689655172413*KNN : 0.8620689655172413 *arbres de décision : 0.8620689655172413*Multilayer Perceptron : 0.06896551724137931 *Adaboost : 0.7241379310344828*Random Forest : 0.8275862068965517

je recommande d'utiliser les 3 algorithmes : SVM ,DecisionTree ou Naïve Bayes qui donne le meilleur score 0.8620689655172413 .

Partie 3 : Regression lineaire

In [6]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

In [7]:

```
df=pd.read_excel('maison_bizertev2.xlsx')
```

In [8]:

```
df.head()
```

Out[8]:

	localisation	surface	nombre.chambre	prix
0	corniche	237	3	650
1	centre.ville	50	1	100
2	bhira	100	2	240
3	manzel.bourguiba	130	3	200
4	sejnen	170	3	180

In [9]:

```
X=df.iloc[:,1:3]
```

In [12]:

```
y=df.iloc[:,-1]
```

```
X prend les caracteristques
y cible prix
```

In [13]:

```
y
```

Out[13]:

```
0     650
1     100
2     240
3     200
4     180
5     570
6     177
7     120
8     133
9      83
10    600
11    100
12    300
13    110
14     58
15    140
16     90
17     80
18     79
```

```
19    180
20     80
21    700
22    150
23    166
24     90
25    100
26    250
27    100
28    120
29     85
30    140
31    200
32    166
33    105
34    110
35    120
36     90
37    780
38    180
39     90
40     60
41    120
42    230
43     70
44    190
45    123
Name: prix, dtype: int64
```

In [14]:

```
regress=LinearRegression().fit(X,y)
```

pour céer une regression avec les deux attributs X et y

In [15]:

```
from sklearn.metrics import mean_squared_error,r2_score
```

importation de deux méthodes :
* moindres carré
* r2 score

In [16]:

```
regress.coef_
```

Out[16]:

```
array([ 2.53932692, 13.59972766])
```

In [17]:

```
regress.intercept_
```

Out[17]:

```
-52.79700701924344
```

In [18]:

```
regress.predict(X)
```

Out[18]:

```
array([589.82265716,  87.76906688, 228.33514079, 318.11467619,
       419.68775319, 279.12167929, 156.33089386,  75.07243226,
```

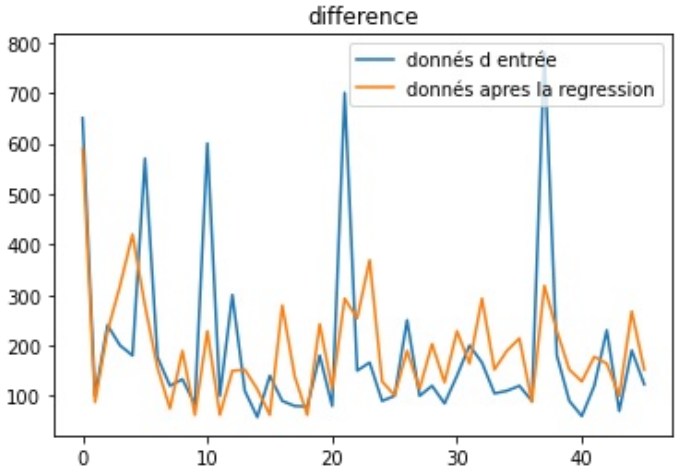
```
       189.34214388,  62.37579764, 228.33514079,  62.37579764,
       149.61600612, 152.15533304, 113.16233613,  62.37579764,
       279.12167929, 138.55560538,  62.37579764, 241.93486845,
       114.06542917, 292.72140694, 253.72841004, 368.90121469,
       128.39829768, 101.36879454, 189.34214388, 114.06542917,
       202.94187154, 126.76206379, 228.33514079, 163.94887463,
       292.72140694, 152.15533304, 189.34214388, 214.00227227,
        87.76906688, 318.11467619, 228.33514079, 152.15533304,
       128.39829768, 177.54860229, 163.94887463, 100.46570151,
       267.32813769, 152.15533304])
```

In [25]:

```
plt.plot(y,label='donnés d entrée')
plt.plot(regress.predict(X),label='donnés apres la regression')
plt.title('difference')
plt.legend(loc=0)
```

Out[25]:

```
<matplotlib.legend.Legend at 0xa5b598ae50>
```



la figure "difference", montre les données d'entrée en bleu et les données créer aprés la regression

In [27]:

```
mean_squared_error(y,regress.predict(X))
```

Out[27]:

```
19508.57823327489
```

In [28]:

```
r2_score(y,regress.predict(X))
```

Out[28]:

```
0.35342071621440785
```

Puisque le score est très faible, il existe deux solutions pour augmenter la performance de ce dernier:
* ajouter plus des donnés d'entrée
* ajouter des features

In [29]:

```
df2=pd.read_excel('maison_bizertev2.xlsx')
```

In [30]:

```
X=df2.iloc[:,1:3]
```

In [31]:

```
y=df2.iloc[:,-1]
```

In [33]:

```
y
```

Out[33]:

```
0     650000
1     100000
2     240000
3     200000
4     180000
5     570000
6     177000
7     120000
8     133000
9      83000
10    600000
11    100000
12    300000
13    110000
14     58000
15    140000
16     90000
17     80000
18     79000
19    180000
20     80000
21    700000
22    150000
23    166000
24     90000
25    100000
26    250000
27    100000
28    120000
29     85000
30    140000
31    200000
32    166000
33    105000
34    110000
35    120000
36     90000
37    780000
38    180000
39     90000
40     60000
41    120000
42    230000
43     70000
44    190000
45    123000
Name: prix, dtype: int64
```

In [34]:

```
regress=LinearRegression().fit(X,y)
```

In [35]:

```
from sklearn.metrics import mean_squared_error,r2_score
```

In [36]:

```
regress.coef_
```

Out[36]:

```
array([ 2539.32692493, 13599.72765721])
```

In [37]:

```
regress.intercept_
```

Out[37]:

```
-52797.00701924341
```

In [38]:

```
regress.predict(X)
```

Out[38]:

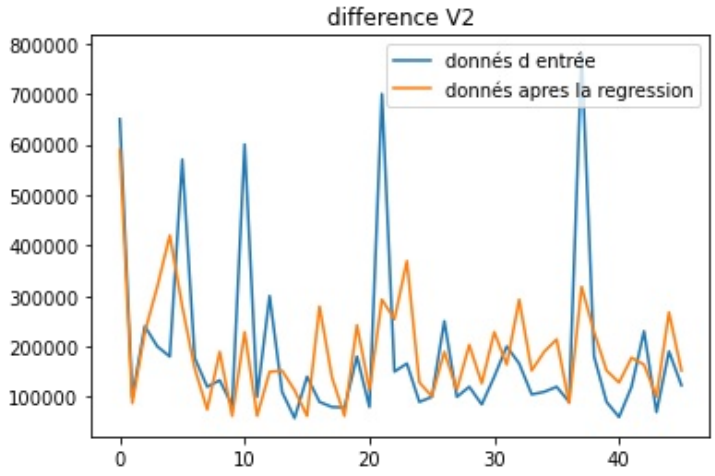
```
array([589822.65716126,  87769.06688456, 228335.14078837, 318114.67619354,
       419687.75319082, 279121.67928701, 156330.89385773,  75072.4322599 ,
       189342.14388184,  62375.79763524, 228335.14078837,  62375.79763524,
       149616.00611548, 152155.33304041, 113162.33613388,  62375.79763524,
       279121.67928701, 138555.6053832 ,  62375.79763524, 241934.86844558,
       114065.42916643, 292721.40694422, 253728.41003769, 368901.21469218,
       128398.29768348, 101368.79454177, 189342.14388184, 114065.42916643,
       202941.87153905, 126762.06379109, 228335.14078837, 163948.87463252,
       292721.40694422, 152155.33304041, 189342.14388184, 214002.27227133,
        87769.06688456, 318114.67619354, 228335.14078837, 152155.33304041,
       128398.29768348, 177548.60228973, 163948.87463252, 100465.70150922,
       267328.1376949 , 152155.33304041])
```

In [39]:

```
plt.plot(y,label='donnés d entrée')
plt.plot(regress.predict(X),label='donnés apres la regression')
plt.title('difference V2')
plt.legend(loc=0)
```

Out[39]:

```
<matplotlib.legend.Legend at 0xa5b4d80730>
```



In [40]:

```
mean_squared_error(y,regress.predict(X))
```

Out[40]:

```
19508578233.27489
```

In [41]:

```
r2_score(y,regress.predict(X))
```

Out[41]:

```
0.3534207162144076
```

Le même score que le premier donc je dois changer le fichier avec des autres données .

In [42]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

In [43]:

```
fichier2=pd.read_excel('maison_tunisie.xlsx')
```

In [44]:

```
fichier2.headad()
```

Out[44]:

	localisation	nombre de chambre	espace	prix
0	Hammamet	3	130	291340000
1	Manouba	1	95	132050000
2	Manouba	1	54	97730000
3	Manouba	2	98	161300000
4	Manouba	3	116	205010000

In [45]:

```
X=fichier2.iloc[:,1:3]
```

In [46]:

```
X
```

Out[46]:

	nombre de chambre	espace
0	3	130
1	1	95
2	1	54
3	2	98
4	3	116
5	4	147
6	1	71
7	2	90
8	3	110
9	2	77
10	3	94
11	1	77
12	2	92
13	3	119
14	1	71
15	3	137
16	2	90
17	3	107

	nombre de chambre	espace
18	2	91
19	3	108
20	2	76
21	3	104
22	1	65
23	2	95
24	3	116

In [47]:

```
y=fichier2.iloc[:,-1]
```

In [48]:

```
y
```

Out[48]:

```
0     291340000
1     132050000
2      97730000
3     161300000
4     205010000
5     343740000
6      84780000
7     108250000
8     135420000
9      86020000
10    105570000
11    143190000
12    170510000
13    214330000
14    139560000
15    259070000
16    124510000
17    148490000
18    140450000
19    165480000
20    103950000
21    142590000
22     96480000
23    142570000
24    174850000
Name: prix, dtype: int64
```

In [49]:

```
regression=LinearRegression().fit(X,y)
```

In [50]:

```
from sklearn.metrics import mean_squared_error,r2_score
```

In [51]:

```
regression.coef_
```

Out[51]:

```
array([-28151499.10115752,   3386328.13590612])
```

In [52]:

```
regression.intercept_
```

Out[52]:

```
-109402136.8234821
```

In [53]:

```
regression.predict(X)
```

Out[53]:

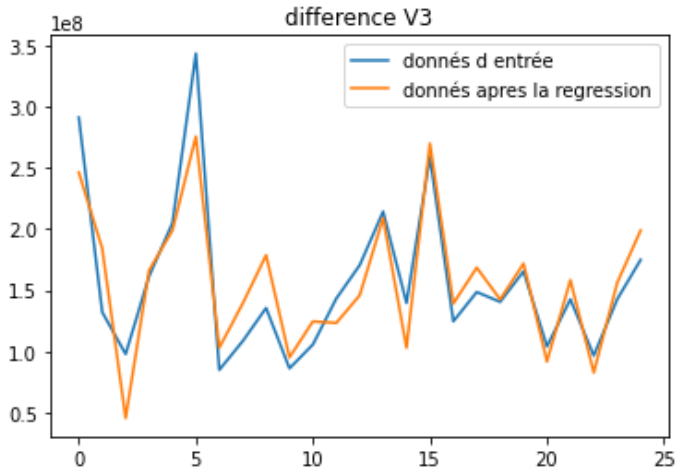
```
array([2.46366024e+08, 1.84147537e+08, 4.53080834e+07, 1.66155022e+08,
       1.98957430e+08, 2.75782103e+08, 1.02875662e+08, 1.39064397e+08,
       1.78639461e+08, 9.50421314e+07, 1.24458211e+08, 1.23193631e+08,
       1.45837053e+08, 2.09116414e+08, 1.02875662e+08, 2.70070320e+08,
       1.39064397e+08, 1.68480476e+08, 1.42450725e+08, 1.71866805e+08,
       9.16558033e+07, 1.58321492e+08, 8.25576929e+07, 1.55996038e+08,
       1.98957430e+08])
```

In [55]:

```
plt.plot(y,label='donnés d entrée')
plt.plot(regression.predict(X),label='donnés apres la regression')
plt.title('difference V3')
plt.legend(loc=0)
```

Out[55]:

```
<matplotlib.legend.Legend at 0xa5b5861910>
```



In [56]:

```
mean_squared_error(y,regression.predict(X))
```

Out[56]:

```
810363263799297.6
```

In [57]:

```
r2_score(y,regression.predict(X))
```

Out[57]:

```
0.7936630073931993
```

On peut constater que le score de ce fichier est très élevé par rapport aux scores précédents.